Bioinformatics Advance Access published February 4, 2015

Bioinformatics, 2015, 1–3
doi: 10.1093/bioinformatics/btu864
Advance Access Publication Date: 18 January 2015
Applications Note

Systems biology

Functional Gene Networks: R/Bioc package to generate and analyse gene networks derived from functional enrichment and clustering

Sara Aibar, Celia Fontanillo†, Conrad Droste and Javier De Las Rivas*

Bioinformatics and Functional Genomics Research Group, Cancer Research Center (Consejo Superior de Investigaciones Científicas, Universidad de Salamanca and Instituto de Investigación Biomédica de Salamanca, CSIC/USAL/IBSAL), Salamanca, Spain

*To whom correspondence should be addressed.
†Present address: Celgene Institute for Translational Research Europe (CITRE), Sevilla, Spain
Associate Editor: Jonathan Wren

Received on June 20, 2014; revised on December 16, 2014; accepted on December 29, 2014

Abstract

Summary: Functional Gene Networks (*FGNet*) is an R/Bioconductor package that generates gene networks derived from the results of functional enrichment analysis (FEA) and annotation clustering. The sets of genes enriched with specific biological terms (obtained from a FEA platform) are transformed into a network by establishing links between genes based on common functional annotations and common clusters. The network provides a new view of FEA results revealing gene modules with similar functions and genes that are related to multiple functions. In addition to building the functional network, *FGNet* analyses the similarity between the groups of genes and provides a distance heatmap and a bipartite network of functionally overlapping genes. The application includes an interface to directly perform FEA queries using different external tools: *DAVID*, *GeneTerm Linker*, *TopGO* or *GAGE*; and a graphical interface to facilitate the use.
Availability and implementation: *FGNet* is available in Bioconductor, including a tutorial. URL: http://bioconductor.org/packages/release/bioc/html/FGNet.html
Contact: jrivas@usal.es
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Due to the increasing number of omic studies, efficient functional analysis of large lists of genes or proteins is essential to understand the biological processes in which they are involved. Functional enrichment analysis (FEA) is the most popular bioinformatic methodology to obtain significant functional information from sets of cooperating genes. FEA methods search in biological databases (such as Gene Ontology and KEGG pathways, among others) and use statistical testing to find biological terms and functional annotations that are significantly enriched in a list of genes. However, in most cases the results of these analyses are very long lists of biological terms associated to genes that are difficult to digest and interpret. Some tools cluster the FEA results, like *DAVID-FAC* (Huang *et al.*, 2009) and *GeneTerm Linker* (Fontanillo *et al.*, 2011), but their output is provided as large tables and there are not many tools to integrate and visualize these results. Here we present Functional Gene Networks (*FGNet*), an R/Bioconductor package that uses FEA results to perform network-based analyses and visualization. The main network is built by establishing links between genes annotated to similar functional terms. In this way, *FGNet* generates and provides a network representing the links and associations between the clusters of genes and enriched terms. The network summarizes and facilitates the interpretation of the biological processes significantly enriched in the initial list of genes, revealing important information such as: distance and overlap

between clusters, identification of modules and hubs. The tool can also help to disclose new associations among genes cooperating in hidden biological processes not annotated yet, which can be revealed by the topology of the functional network.

2 Methods

2.1 Input: functional enrichment and clustering

FGNet builds functional networks based on the groups obtained from clustering *gene-term sets* (*gtsets*, genes and terms associated by an enrichment p-value) returned by a FEA. The package includes an interface to do queries with gene lists using four FEA tools: *DAVID* with *Functional Annotation Clustering* (that returns clustered *gtsets*, Cl); *GAGE* (that also provides clusters) (Luo *et al.*, 2009); *GeneCodis* with *GeneTerm Linker* (that returns metagroups, Mg) and *TopGO* (that only returns *gtsets*) (Alexa *et al.*, 2010). The package can be also applied to the results from other EA tools, as long as the input results are transformed into tables of genes and associated terms.

2.2 Construction of the functional network

The functional network is built based on the analysis of all the *gtsets* provided by the FEA tool. These sets allow to generate a boolean matrix M of *genes* by *gtsets*, in which each element $m_{g,s} = 1$ if gene g is in set s. This membership matrix is then transformed into an adjacency matrix A $n \times n$; being n the total number of genes and $a_{i,j}$ the number of *gtsets* s in which a gene-pair is included: $a_{i,j} = \sum_s (m_{i,s} \times m_{j,s})(1 - \delta_{i,j})$, where δ is a *Kronecker* delta ($\delta_{i,j} = 1$ if $i = j$, $\delta_{i,j} = 0$ if $i \neq j$). This adjacency matrix is used to generate the functional network by establishing a weighted link between each pair of genes (g_i, g_j) in which $a_{i,j} \neq 0$. Finally, the clustering of *gtsets* provided by the FEA tool is used to generate a second genes' adjacency matrix with the number of common clusters/metagroups (Fig. 1A), that is used to define and allocate gene groups. The network produced is provided as an *igraph* object for further analysis, and can be exported to other network-based tools like *Cytoscape*.

2.3 Visualization and plots of the functional network

The main plot of the network presents the functionally associated genes (Fig. 1B). Edges link the genes that are in the same *gtsets*. Nodes within the same Cl/Mg are placed together using a force-directed *Fruchterman–Reingold* layout, within a common background colour. Genes in only one Cl/Mg are plotted with the colour of such group, while genes that are included in more than one Cl/Mg are left white.

2.4 Analysis of functional modules in the network

To facilitate the analysis and quantification of the modules and the overlap between groups, *FGNet* also provides a distance matrix and a heatmap (Fig. 1C), plus an intersection network (Fig. 1D). The distance matrix is calculated based on the pairwise binary distance in the adjacency matrix of common Cls/Mgs. These distances are analysed by hierarchical average linkage and plotted as a heatmap that reveals the proximity and similarity between the groups of genes (Cls/Mgs). The intersection network is a bipartite network which includes only the genes associated to several Cls/Mgs (white nodes in Fig. 1B,D), showing their connectivity to such Cls/Mgs. This intersection network facilitates the identification of *multifunctional* genes. (For more details see *FGNet* documentation in Bioconductor).

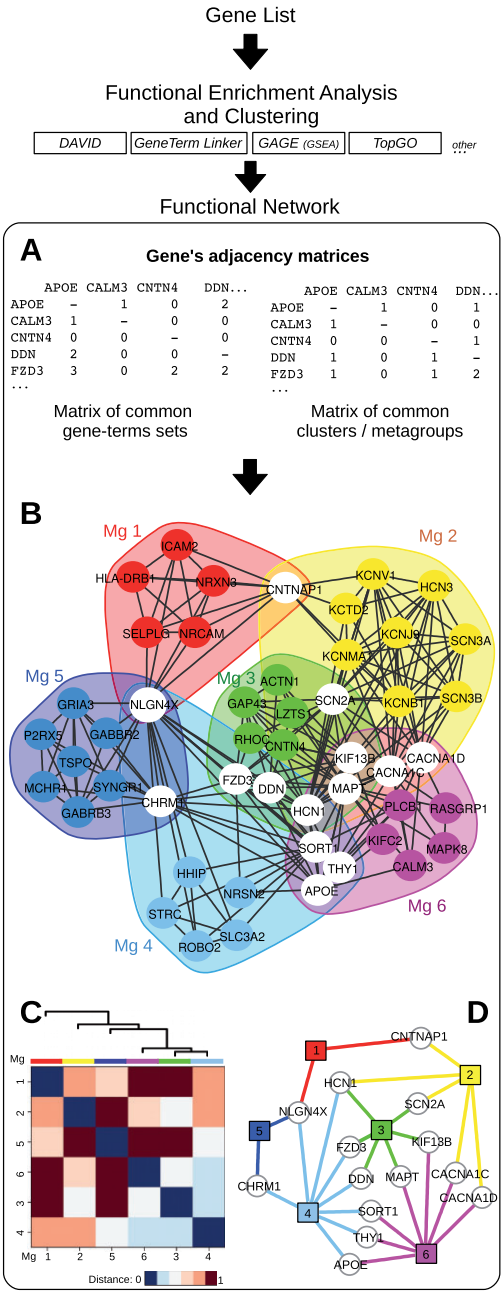


Fig. 1. Schematic workflow. A query gene list is analysed through a FEA tool and the generated *gene-term sets* are used to build: (**A**) gene's adjacency matrices; (**B**) a functional network (general view); (**C**) a distance heatmap and (**D**) an intersection network (to highlight multifunctional genes)

3 Example of use

We applied the method to several datasets and confirmed that the functional network greatly facilitates the analysis of enrichment results. Figure 1 shows the results of *FGNet* for a list of 175 genes

differentially expressed in human samples of entorhinal cortex neurons from Alzheimer's disease (AD) patients (obtained from Gene Expression Omnibus database, GEO: dataset GSE4757). Performing a FEA through *GeneTerm Linker*, we obtained six meta-groups that we labelled according to their main annotations: (Mg1) cell adhesion; (Mg2) voltage-gated ion/potassium channels; (Mg3) axon and cell projection; (Mg4) dendrite and neuronal cell body; (Mg5) synaptic neuroactive ligand-receptor interaction and (Mg6) MAPK signaling and Alzheimer. The network of these six Mgs (Fig. 1B) provides a global overview of the functionally overlapping genes and allows to identify hub genes that interconnect groups. For example, CNTNAP1 and NLGN4X appear as hubs in Mg1. CNTNAP1 (that regulates distribution of K+ channels) links Mg1 and 2; and NLGN4X (that facilitates synaptic neurotransmission) links Mg1 with 4 and 5. NLGN4X is the gene with highest betweenness centrality in this network. Another important hub is APOE, recently associated to Alzheimer. The distance matrix (Fig. 1C) allows to quantify the similarity between gene groups, showing that the closest Mgs are 3, 4 and 6, sharing eight nodes. This is also presented in the intersection network (Fig. 1D). Finally, the functional network can reveal further information about some Mgs. For example, if a Mg shares many genes with several other Mgs, it will indicate that such Mg brings the most common features that define the studied biological state. This is the case for Mg6, which, in fact, is annotated to Alzheimer's Disease.

Funding

This work was supported by the "Accion Estrategica en Salud" (AES) of the "Instituto de Salud Carlos III" (ISCiii) from the Spanish Government (projects granted to J.D.L.R.: PS09/00843 and PI12/00624); and by the "Consejeria de Educación" of the "Junta Castilla y Leon" (JCyL) and the European Social Fund (ESF) with grants given to S.A. and C.D.

Conflict of Interest: none declared.

References

Alexa,A. and Rahnenfuhrer,J. (2010). topGO: Enrichment analysis for Gene Ontology. R package version 2.18.0.

Fontanillo,C. *et al.* (2011). Functional analysis beyond enrichment: non-redundant reciprocal linkage of genes and biological terms. *PLoS One,* **6**, e24289.

Huang,D. *et al.* (2009). Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources. *Nat. Protoc.*, **4**, 44–57.

Luo,W. *et al.* (2009). GAGE: generally applicable gene set enrichment for pathway analysis. *BMC Bioinformatics,* **10**, 161.

Aditional files

Aditional file 1: FGNet package vignette

Also available at *Bioinformatics* online and *Bioconductor.*

URLs:

http://bioinformatics.oxfordjournals.org/lookup/suppl/doi:10.1093/bioinformatics/btu864/-/DC1

http://bioconductor.org/packages/release/bioc/vignettes/FGNet/inst/doc/FGNet.html

FGNet
Functional Gene Networks derived from biological enrichment analyses

Sara Aibar, Celia Fontanillo, Conrad Droste, and Javier De Las Rivas

Bioinformatics and Functional Genomics Group
Centro de Investigacion del Cancer (CiC-IBMCC, CSIC/USAL)
Salamanca - Spain

December 10, 2014

Version: 3.0

Contents

1 Introduction to FGNet

FGNet allows to perform a Functional Enrichment Analysis (FEA) on a list of genes or expression set, and transform the results into networks. The resulting functional networks provide an overview of the biological functions of the genes/terms, and allows to easily see links between genes, overlap between clusters, finding key genes, etc.

FGNet takes as input a query list of genes selected by the user, and builds and displays networks of genes based in the existence of common functional terms that are enriched in certain subsets of genes of the list. By doing this, the tool allows to disclose groups/clusters of genes that have similar annotations and so they may have similar biological function in the cell. The discovery of molecular machines or functional modules within the cell (i.e. genes or proteins that work together to perform a biological process in the cells) is essential in modern molecular medicine and systems biology, because many times we do not know which are the gene partners playing in the same roles in a pathological state. FGNet is a tool that helps to create functional connections between different genes/proteins based on annotations. By grouping similar, redundant and homogeneous annotation content from the same or different biological resources into gene-term groups, the biological interpretation of large gene lists moves from a gene centric approach (where each gene is independent) to a functional-module centric approach (where the genes are interconnected). In this way, FGNet can provide a better representation of complex biological processes and reveal associations between genes.

Biological functional analysis

After obtaining a list of genes or proteins from an experiment or omic studies (microarrays, RNAseq, mass spectrometry, etc), the next step is usually to perform a functional analysis of the genes to search for the biological functions or processes in which they are involved. In order to facilitate the analysis of large lists of genes, multiple functional enrichment tools have been developed. These tools search for the genes in biological databases (i.e. GO, Kegg, Interpro), and test whether any biological annotations are over-represented in the query gene list compared to what would be expected in the whole population. However, the raw output from a functional enrichment analysis often provides dozens or hundreds of terms, and it still requires a lot of time and attention to go through the whole list of genes and annotations. A way to simplify this task is grouping genes and terms which often appear together and create associated networks: the Functional Networks.

FGNet builds the functional networks, based on data from a previous functional enrichment analysis (FEA). The package provides the functions to perform the FEA through four specific tools:

- **DAVID** with Functional Annotation Clustering (DAVID-FAC), which measures relationships among annotation terms based on their co-association with subsets of genes within the query gene list (Huang et al.). This type of clustering mostly results in groups of highly related terms, such as synonymous annotations from different annotation spaces (i.e. term “glycolysis” in KEGG and GO-BP), which also share most of their genes. This tool provides great coverage but does not avoid redundant terms and very general terms (like “signal transduction” or “regulation of transcription” that correspond to specific terms in Gene Ontology, GO).

- **GeneTerm Linker**, a post-enrichment tool, which focuses on clearing and sorting the results from a previous modular enrichment analysis. This is achieved by filtering general terms with low information content (i.e. *cellular process* or *protein binding*) and redundant annotations (i.e. *metabolic process* and *primary metabolic process*). The remaining gene-term sets are grouped into `metagroups` based on their shared genes and terms (using a reciprocal linkage approach) (Fontanillo et al.).
- **TopGO** (Alexa et al.), an enrichment analysis tool based on Gene Ontology (GO) that tests GO terms while accounting for the topology of the GO graph to eliminate local similarities and dependencies between GO terms. TopGO does not provide clusters, and therefore the functional network is built using only the gene-term sets. TopGO can be applied off-line.
- **GAGE** (Luo et al.), a gene set enrichment analysis (GSEA) tool. It searches for functional enrichment in gene sets (i.e. KEGG pathways, Reactome, GO) and allows including a signal value -like expression changes- to rank the genes and then to identify the enrichment in functional terms that are altered (i.e. changed in genes UP and DOWN) or altered consistently in one direction (UP or DOWN). GAGE also clusters the resulting enriched gene-term sets and can be applied off-line.

To build the network based on other *other tools*, the raw output should be saved into a text file which contains the enriched terms and their genes. (For more details see function `format_results()`).

Functional network

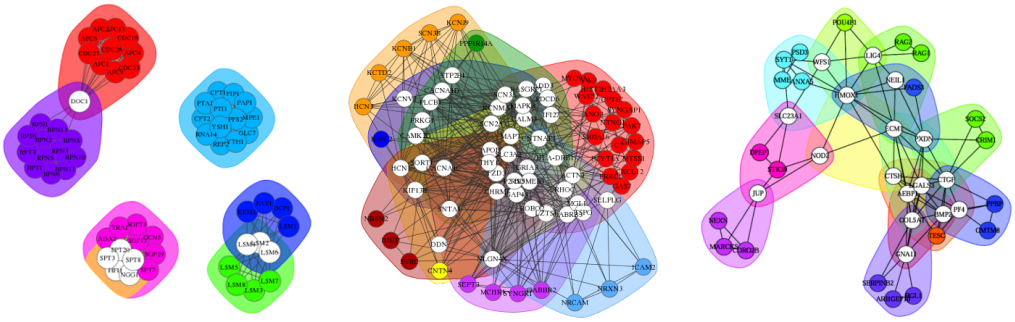
The **functional network** is the representation of the results from a functional enrichment analysis.

In the **default** network, all the nodes of the network are of the same type, i.e. genes OR terms, which are linked to each other if they are in the same gene-term set. In the plot, the genes/terms in the same groups (metagroups or clusters) are surrounded by a common background color.

In the **bipartite** network, the nodes are of two types, allowing to link the genes or terms, with the clusters they belong to. This network, can be built as an *intersection network* , a simplified functional network where all the genes/terms that belong to only one metagroup are clustered into a single node. This simplified network contains only the nodes in several groups.

In addition to the networks, FGNet also provides a few functions for further analysis. These functions allow to get a **distance matrix** , which represents the similarity between the groups based on the genes they share with each other (binary distance), and the distribution of **degree and betweenness** within the network and subnetworks, in order to find the most important genes (hubs).

All these functionalities can be accessed directly through the appropriate functions or the graphical user interface (GUI). In addition, FGNet also allows to generate an **HTML report** with an overview of these plots and analyses for a specific gene list.



Examples of funtional network for different analyses.

2 Installation

To install *FGNet* from *Bioconductor*, type in your R console:

```
source("http://bioconductor.org/biocLite.R")
biocLite("FGNet")
```

To reduce system requirements, only the minimum packages are required to execute FGNet. However, there are several functionalities that require further packages. i.e. the Graphical User Interface (GUI) requires "RGtk2", the FEA analyses might require "RDAVIDWebService", "gage", "topGO" or some annotation packages... etc.

To make sure all FGNet functionalities are available, install the following packages:

```
biocLite(c("RGtk2", "RCurl",
    "RDAVIDWebService", "gage", "topGO", "KEGGprofile",
    "GO.db", "KEGG.db", "reactome.db", "org.Sc.sgd.db"))
```

3 Creating a network from a list of genes/proteins

To generate a functional network with FGNet:

1. Perform a Functional Enrichment Analysis (FEA) on a list of genes or expression set.

FEA tool	Online?	Input	Annotations
DAVID	Yes	Gene list	Many
Gene-Term Linker	Yes	Gene list	GO, KEGG, Interpro
TopGO	No	Gene list	GO
GAGE (GSEA)	No	Expression set	Any gene set

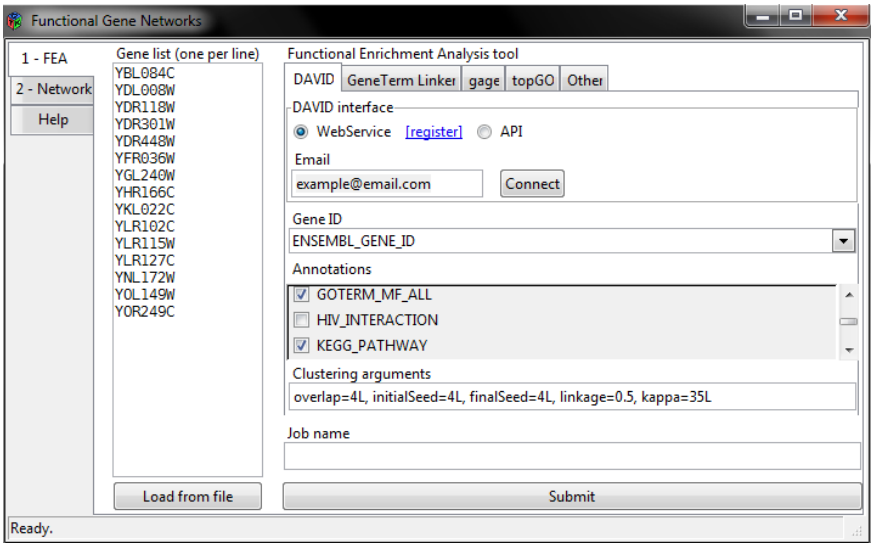
2. Create an HTML report with multiple views of the networks and analyses.
3. Personalize or analyze an specific network.

These steps are integrated into the Graphical User Interface (GUI), which provides access to the main functionalities of FGNet.

3.1 Graphical User Interface (GUI)

The Graphical User Interface (GUI) provides access to most FGNet functionalities in Windows and Linux (The current version of the GUI is not available for Mac OS X Snow Leopard).
To launch the GUI, type in the R console:

```
library(FGNet)
FGNet_GUI()
```



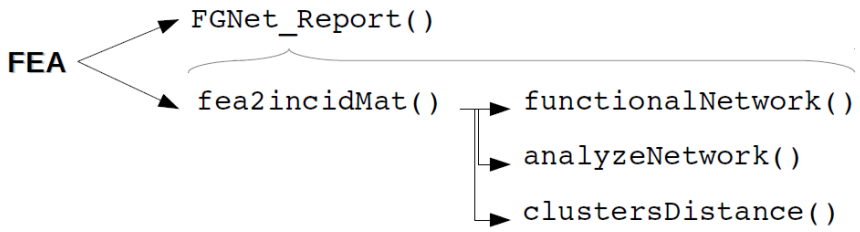
In case you already have a gene list or gene expression from a previous analysis, it is possible to load it directly into the GUI genes field by passing it as argument:

```
geneExpr <- c("YBL084C", "YDL008W", "YDR118W", "YDR301W", "YDR448W",
    "YFR036W", "YGL240W", "YHR166C", "YKL022C", "YLR102C", "YLR115W",
    "YLR127C", "YNL172W", "YOL149W", "YOR249C")
geneExpr <- setNames(c(rep(1,10),rep(-1,5)), geneExpr)

FGNet_GUI(geneExpr)
```

3.2 In R code...

The first step in the workflow is always is to perform a Functional Enrichment Analysis (FEA) on a list of genes or expression set.
Once the FEA is ready, you can proceed to generate the HTML report or the individual network/analyses:

FGNet 7



For help or more details on any functions or their arguments, just set a ? before its name.

```
?FGNet_report
```

3.2.1 Functional Enrichment Analysis (FEA)

Since the arguments required to perform the FEA depends on the tool, there are several FEA functions:

FEA tool	Function	Output group type
DAVID	`fea_david()`	Clusters
TopGO	`fea_topGO()`	No grouping
Gene-Term Linker	`fea_gtLinker()` & `fea_gtLinker_getResults()`	Metagroups
GAGE	`fea_gage()`	Clusters
Other	`format_feaResults()`	

All the FEA functions and `FGNet_report()` save the results in the current working directory.

```
getwd()
```

Here is an example analyzing a gene list with the different tools:

```
genesYeast <- c("ADA2", "APC1", "APC11", "APC2", "APC4", "APC5", "APC9",
    "CDC16",  "CDC23", "CDC26", "CDC27", "CFT1", "CFT2", "DCP1", "DOC1",
    "FIP1", "GCN5", "GLC7", "HFI1", "KEM1", "LSM1", "LSM2", "LSM3",
    "LSM4", "LSM5", "LSM6", "LSM7", "LSM8", "MPE1", "NGG1", "PAP1",
    "PAT1", "PFS2", "PTA1", "PTI1", "REF2", "RNA14", "RPN1", "RPN10",
    "RPN11", "RPN13", "RPN2", "RPN3", "RPN5", "RPN6", "RPN8", "RPT1",
    "RPT3", "RPT6", "SGF11", "SGF29", "SGF73", "SPT20", "SPT3", "SPT7",
    "SPT8", "TRA1", "YSH1", "YTH1")

library(org.Sc.sgd.db)
geneLabels <- unlist(as.list(org.Sc.sgdGENENAME))
genesYeast <- sort(geneLabels[which(geneLabels %in% genesYeast)])

# Optional: Gene expression (1=UP, -1=DW)
genesYeastExpr <- setNames(c(rep(1,28), rep(-1,30)),genesYeast)
```

DAVID
Using DAVID requires internet connection. In addition, we recommend to register at `http://david.abcc.ncifcrf.gov/webservice/register.htm` to perform the queries through its Web Service.

By default, `geneIdType="ENSEMBL_GENE_ID"`. To replace the gene IDs by readable names in the plots and HTML report, use the argument `geneLabels`. To see the gene IDs supported by DAVID's Web Service, use: `getIdTypes(DAVIDWebService$new(email=...))`.

```
feaResults_David <- fea_david(names(genesYeast), geneLabels=genesYeast,
                              email="example@email.com")
?fea_david
```

TopGO
Since TopGO uses local databases, it does not require internet connection.

The results from topGO are provided as individual gene-term sets not grouped into clusters. FGNet treats each *gene-term set* as a single cluster.

```
feaResults_topGO <- fea_topGO(genesYeast,
    geneIdType="GENENAME", organism="Sc")
?fea_topGO
```

Gene-Term Linker
Since the analysis with Gene-Term Linker usually takes several minutes to be ready, the workflow is divided in two steps: (1) sending the analysis request, and (2) retrieving the results:

```
jobID <- fea_gtLinker(geneList=genesYeast, organism="Sc")
?fea_gtLinker
```

once the analysis is ready...

```
jobID <- 3907019
feaResults_gtLinker <- fea_gtLinker_getResults(jobID=jobID, organism="Sc")
```

GAGE
As a GSEA approach, instead of performing the functional enrichment over a gene list, gage requires a raw expression set and the samples to compare:

```
library(gage)
data(gse16873)

# Set gene labels? (they need to have unique identifiers)
library(org.Hs.eg.db)
geneSymbols <- select(org.Hs.eg.db,columns="SYMBOL",keytype="ENTREZID",
    keys=rownames(gse16873))
```

```
geneLabels <- geneSymbols$SYMBOL
names(geneLabels) <- geneSymbols$ENTREZID
head(geneLabels)

# GAGE:
feaResults_gage <- fea_gage(eset=gse16873,
                            refSamples=grep('HN',colnames(gse16873)),
                            compSamples=grep('DCIS',colnames(gse16873)),
                            geneLabels=geneLabels, annotations="REACTOME",
                            geneIdType="ENTREZID", organism="Hs")
?fea_gage
```

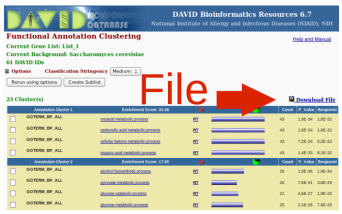
Other tools
To import the results from a functional enrichment analysis performed with other tools, see:

```
?format_results()
```

Web analysis
FGNet can also be applied to an analysis performed at DAVID and GeneTerm Linker web site:

- DAVID: `http://david.abcc.ncifcrf.gov` (Functional Annotation Clustering Tool)
- GeneTerm Linker: `http://gtlinker.cnb.csic.es`

To import these results into FGNet, use DAVID's `download file` or GeneTerm linker's `job ID`, and the functions `format_david()` or `fea_gtLinker_getResults()`:




```
feaResults_David <- format_david(
    "http://david.abcc.ncifcrf.gov/data/download/90128.txt")
feaResults_gtLinker <- fea_gtLinker_getResults(jobID=3907019)
```

3.2.2 HTML report

The HTML report function allows to create a comprehensive report including different views of the Functional Network, the cluster/metagroup legend, and some further statistics directly directly from a gene list.

Here is the code to use `FGNet_report()` with each of the previous examples:

```
FGNet_report(feaResults_David, geneExpr=genesYeastExpr, plotKeggPw=FALSE)
FGNet_report(feaResults_topGO, geneExpr=genesYeastExpr)
FGNet_report(feaResults_gtLinker, geneExpr=genesYeastExpr)
FGNet_report(feaResults_gage)
```

By default, the clusters included in these reports are filtered out to get cleaner results. The default values depend on the tool, and can be modified through FGNet_report arguments:

```
data(FEA_tools)
FEA_tools[,4:6]
```

```
FGNet_report(feaResults_gtLinker, filterThreshold=0.3)
```

```
?FGNet_report
```

3.2.3 Individual networks

After the FEA is ready, it is also possible to generate specific networks rather than the full report. Here is a simple example on how to use `fea2incidMat()` to generate the incidence matrices that represent the networks and plot them. There are more detailed examples on how to edit and explore the networks in sections *editing and creating new networks* (sec. 4) and *filtering and selecting clusters* (sec. 5).

```
feaResults <- feaResults_gtLinker
incidMat <- fea2incidMat(feaResults)
incidMat$metagroupsMatrix[1:5, 1:5]
```

```
##       1 2 3 4 5
## ADA2  0 1 0 0 0
## APC1  0 0 1 0 0
## APC11 0 0 1 0 0
## APC2  0 0 1 0 1
## APC4  0 0 1 0 0
```

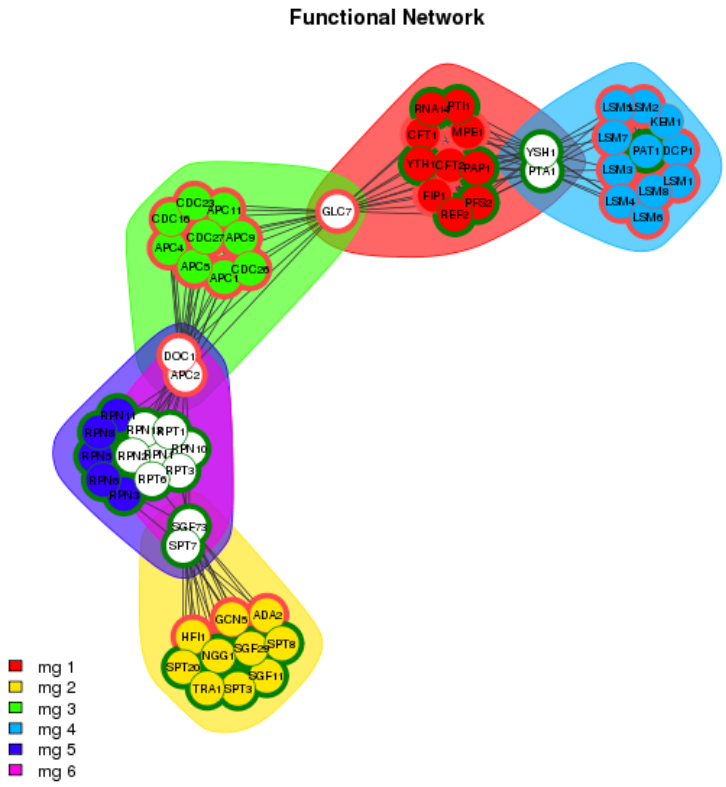
```
incidMat_terms <- fea2incidMat(feaResults, key="Terms")
incidMat_terms$metagroupsMatrix[5:10, 1:5]
```

```
##                                                 1 2 3 4 5
## Chromatin assembly (BP) (GO:0031497)            0 0 1 0 0
## Chromatin modification (BP) (GO:0016568)        0 1 0 0 0
## Cytoplasmic mRNA processing body (CC) (GO:0000932) 0 0 0 1 0
## Enzyme regulator activity (MF) (GO:0030234)     0 0 0 0 1
## Histone acetylation (BP) (GO:0016573)           0 1 0 0 0
## Histone acetyltransferase activity (MF) (GO:0004402) 0 1 0 0 0
```

These incidence matrices can be plotted and analyzed in different ways:

FGNet 11

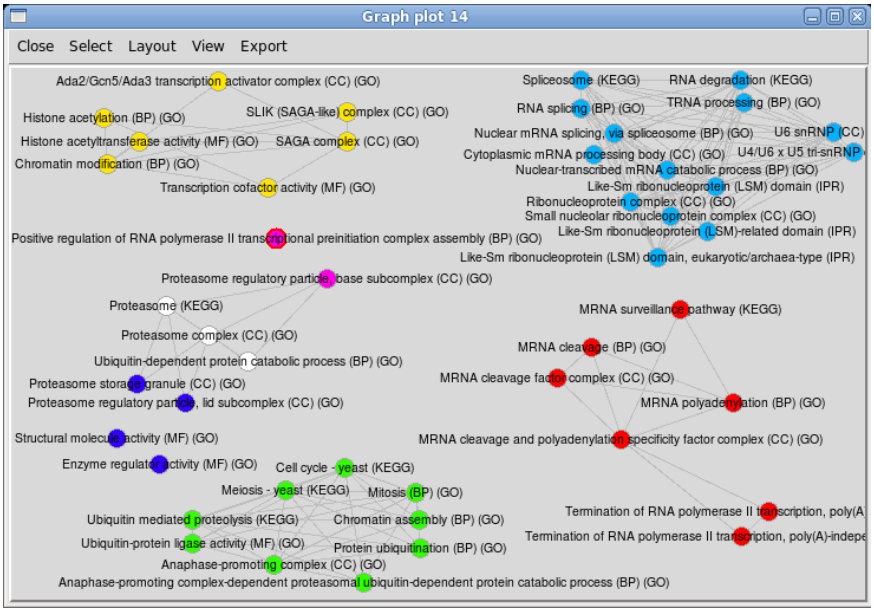
```
functionalNetwork(incidMat, geneExpr=genesYeastExpr,
    plotTitleSub="Default gene view")
```



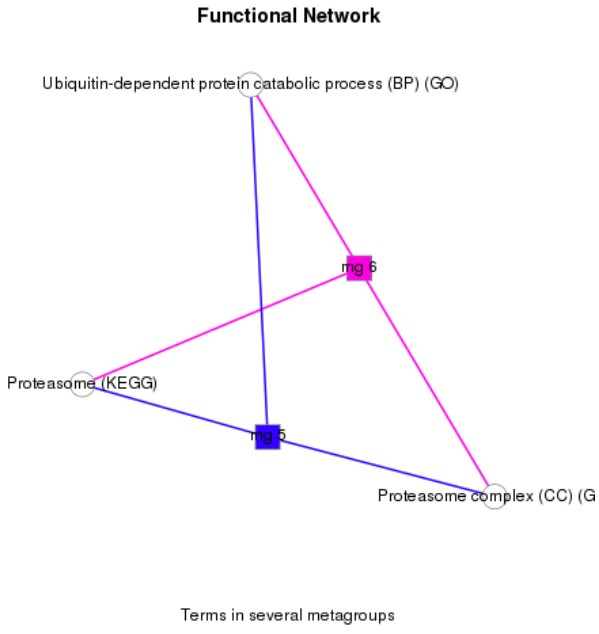

```
getTerms(feaResults)[1]
```

```
## $`Metagroup 1`
##      Terms
## [1,] "MRNA cleavage and polyadenylation specificity factor complex (CC)"
## [2,] "MRNA cleavage (BP)"
## [3,] "MRNA cleavage factor complex (CC)"
## [4,] "MRNA polyadenylation (BP)"
## [5,] "MRNA surveillance pathway"
## [6,] "Termination of RNA polymerase II transcription, poly(A)-coupled (BP)"
## [7,] "Termination of RNA polymerase II transcription, poly(A)-independent (BP)"
```

```
functionalNetwork(incidMat_terms, plotOutput="dynamic")
```




```
functionalNetwork(incidMat_terms, plotType="bipartite",
    plotTitleSub="Terms in several metagroups")
```



4 Editing and creating new networks

In this section we will use the functional analysis of an Alzheimer dataset (GSE4757):

```
jobID <- 1639610
feaAlzheimer <- fea_gtLinker_getResults(jobID=jobID, organism="Hs")
```

The variable `feaAlzheimer` contains the raw results from the functional analysis. The slot `metagroups` could also be `clusters` or missing depending on the FEA tool:

```
names(feaAlzheimer)
```

```
## [1] "queryArgs"    "metagroups"   "geneTermSets" "fileName"
```

```
head(feaAlzheimer$metagroups)
```

To see the terms in each cluster/metagroup use `getTerms()`:

```
getTerms(feaAlzheimer)[3:4]
```

```
## $`Metagroup 3`
##       Terms
##  [1,] "Alzheimer's disease"
##  [2,] "Calcium ion transport (BP)"
##  [3,] "Calcium signaling pathway"
##  [4,] "Calmodulin binding (MF)"
##  [5,] "GnRH signaling pathway"
##  [6,] "Induction of apoptosis by extracellular signals (BP)"
##  [7,] "Long-term potentiation"
##  [8,] "Melanogenesis"
##  [9,] "Neurotrophin signaling pathway"
## [10,] "Salivary secretion"
## [11,] "Tuberculosis"
## [12,] "Vascular smooth muscle contraction"
## [13,] "Wnt signaling pathway"
##
## $`Metagroup 4`
##      Terms
## [1,] "Glutamatergic synapse"
## [2,] "Postsynaptic density (CC)"
## [3,] "Postsynaptic membrane (CC)"
## [4,] "Synapse (CC)"
```

4.1 Incidence matrices

The FEA results should be transformed into incidence matrices to create the network. These matrices are the internal representation of the network: they contain which genes are in each metagroup or cluster and in each gene-term set. Therefore, it is in this step where the main shape of the network is determined.

The function to create the incidence matrices is `fea2incidMat()`. It allows to filter out clusters, decide whether the networks should be gene-based or term-based, establish the groups to link the genes/terms, etc...
We will start the example creating a simple gene-based network:

```
incidMat <- fea2incidMat(feaAlzheimer)
```

```
head(incidMat$metagroupsMatrix)
```

```
##        1 2 3 4 5 6 7 8 9
## ACTN1  1 0 0 0 1 0 0 0 0
## ADD3   1 0 1 0 0 0 0 0 0
## ANO3   1 0 0 0 0 0 0 0 0
## APOE   1 0 0 0 0 1 0 0 1
## ATP2B1 0 0 1 0 0 0 0 0 0
## C7     1 0 0 0 0 0 0 0 0
```

```
incidMat$gtSetsMatrix[1:5, 14:18]
```

```
##        1.14 1.15 1.16 1.17 1.18
## ACTN1     0    1    0    1    1
## ADD3      0    0    0    0    0
## ANO3      0    0    0    0    0
## APOE      0    0    1    0    0
## ATP2B1    0    0    0    0    0
```

To filter or select with metagroups to show, use the arguments `filterAttribute`, `filterOperator` and `filterThreshold`. `filterAttribute` should be a column from the `feaAlzheimer$clusters` or `feaAlzheimer$metagroups` data frames. The recommended filters for each tool can be seen in the object `FEA_tools`, which contains the default filters when generating the HTML report:

```
data(FEA_tools)
FEA_tools[,4:6]
```

```
incidMatFiltered <- fea2incidMat(feaAlzheimer,
    filterAttribute="Silhouette Width", filterOperator="<", filterThreshold=0.2)
```

To see which metagroups/clusters have been filtered out and will not be shown in the networks:

```
incidMatFiltered$filteredOut
```

For more on selecting and filtering groups see section 5. To build the networks based on terms, use the argument `key="Terms"`.

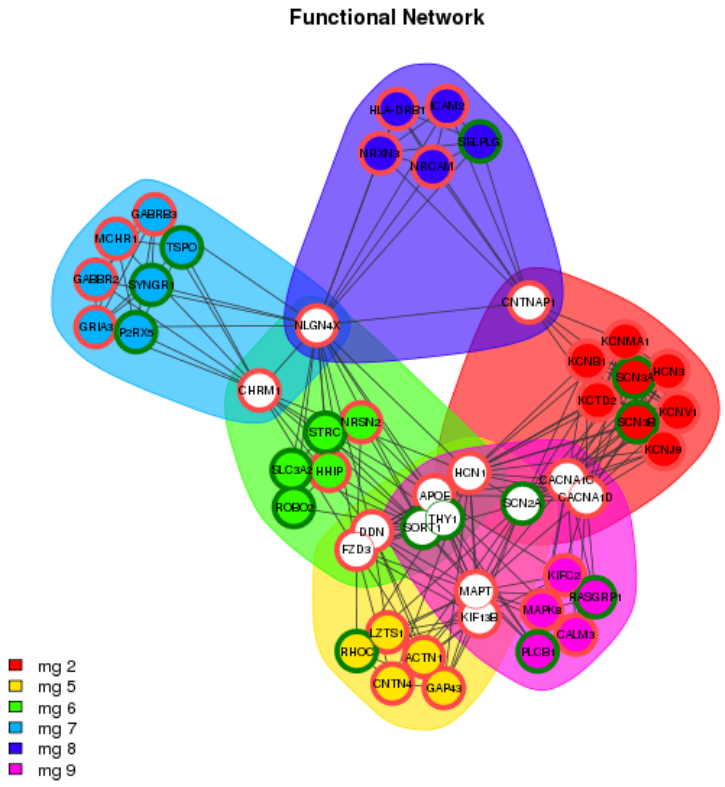
4.2 Functional network

The function `functionalNetwork()` generates and plots the networks. In case there is available expression data, it can be used for representation in this step:

```
# (Fake expression data)
genesAlz <- rownames(incidMat$metagroupsMatrix)
genesAlzExpr <- setNames(c(rep(1,50), rep(-1,27)), genesAlz)
```

The `default` plot will plot all the genes/terms in the network, and will return the networks as igraph objects and matrices in an invisible list. The argument `keepColors` determine whether the colors should be consistent, taking into account the filtered groups, or restarted:

```
fNw <- functionalNetwork(incidMatFiltered, geneExpr=genesAlzExpr, keepColors=FALSE)
```



By setting the parameter `plotOutput="dynamic"` instead of an static plot, it will create an interactive one. By setting `plotOutput="none"`, it is possible to produce only the network without plotting.

```
functionalNetwork(incidMatFiltered, geneExpr=genesAlzExpr, plotOutput="dynamic")
fNw <- functionalNetwork(incidMatFiltered, plotOutput="none")
```

Since the returned networks are `iGraph` objects, they can be used or analyzed as such:

```
names(fNw)
```

```
## [1] "iGraph" "adjMat"
```

```
names(fNw$iGraph)
```

```
## [1] "commonClusters" "commonGtSets"
```

```
library(igraph)
clNw <- fNw$iGraph$commonClusters
clNw
```

```
## IGRAPH UN-- 49 334 --
## + attr: name (v/c)
```

```
vcount(clNw)
ecount(clNw)
sort(betweenness(clNw), decreasing=TRUE)[1:10]
igraph.to.graphNEL(clNw)
```

In dynamic plots (`tkplot`) it is not possible to draw the metagroup background. However, you can save the layout of a dynamic network, and plot it as static using the argument `vLayout`:

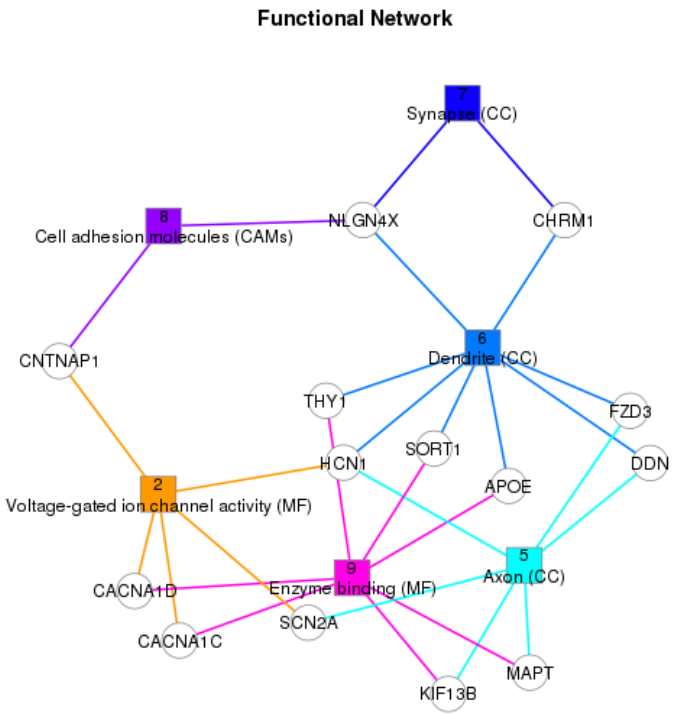
```
functionalNetwork(incidMatFiltered, plotOutput="dynamic")
# Modify the layout...
saveLayout <- tkplot.getcoords(1)    # tkp.id (ID of the tkplot window)
functionalNetwork(incidMatFiltered, vLayout=saveLayout)
```

FGNet 17

4.3 Bipartite and intersection network

The default `bipartite` version of the functional network plots the *intersection network*: a simplified functional network, containing only the nodes in several metagroups and the metagroups they belong to. In this network, metagroup nodes (the coloured nodes) can be seen as a cluster of all the genes/proteins that belong only to that metagroup:

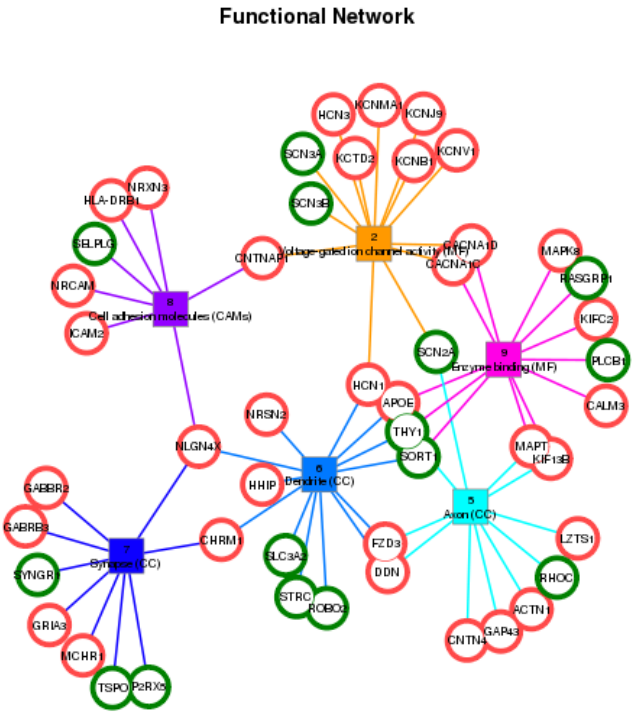
```
mgKeyTerm <- keywordsTerm(getTerms(feaAlzheimer),
    nChar=100)[-c(as.numeric(incidMatFiltered$filteredOut))]
functionalNetwork(incidMatFiltered, plotType="bipartite", legendText=mgKeyTerm)
```



FGNet 18

To plot a full bipartite network including all the nodes, just set `keepAllNodes=TRUE`:

```
functionalNetwork(incidMatFiltered, geneExpr=genesAlzExpr, plotType="bipartite",
    keepAllNodes=TRUE, plotTitleSub="Bipartite network will all nodes",
    legendText=mgKeyTerm)
```



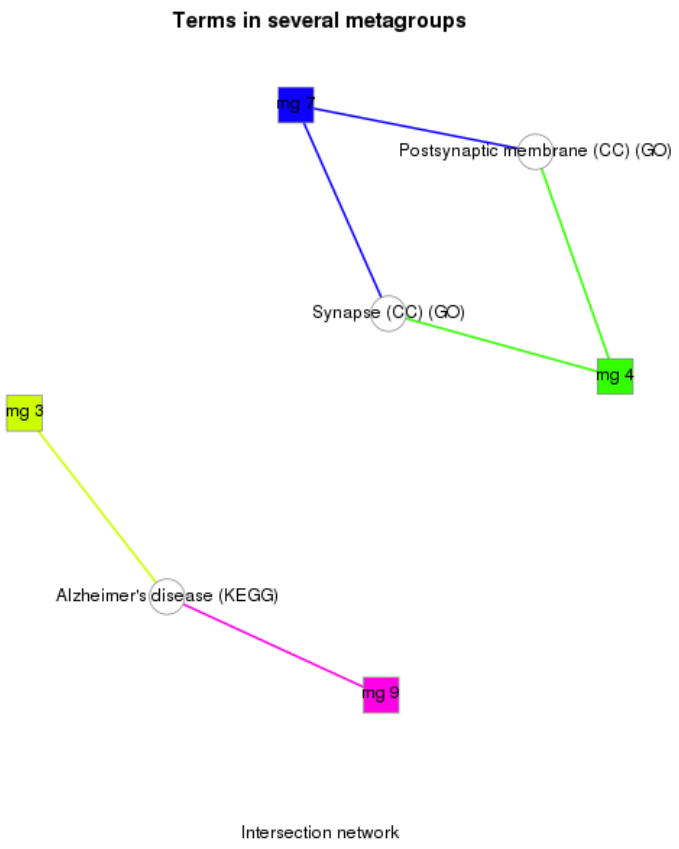
Bipartite network will all nodes

4.4 Terms networks

In the same way we have built networks to explore the relationship between genes, the same approach can be used to explore the relationship between the biological terms in the enrichment analysis. i.e. to see which biological terms are usually associated, or locate which terms are in several groups. To do so, build the incidence matrices based on terms instead of genes using the argument `key="Terms"`.

```
incidMatTerms <- fea2incidMat(feaAlzheimer, key="Terms")
```

```
functionalNetwork(incidMatTerms, plotType="bipartite",
    plotTitle="Terms in several metagroups")
```



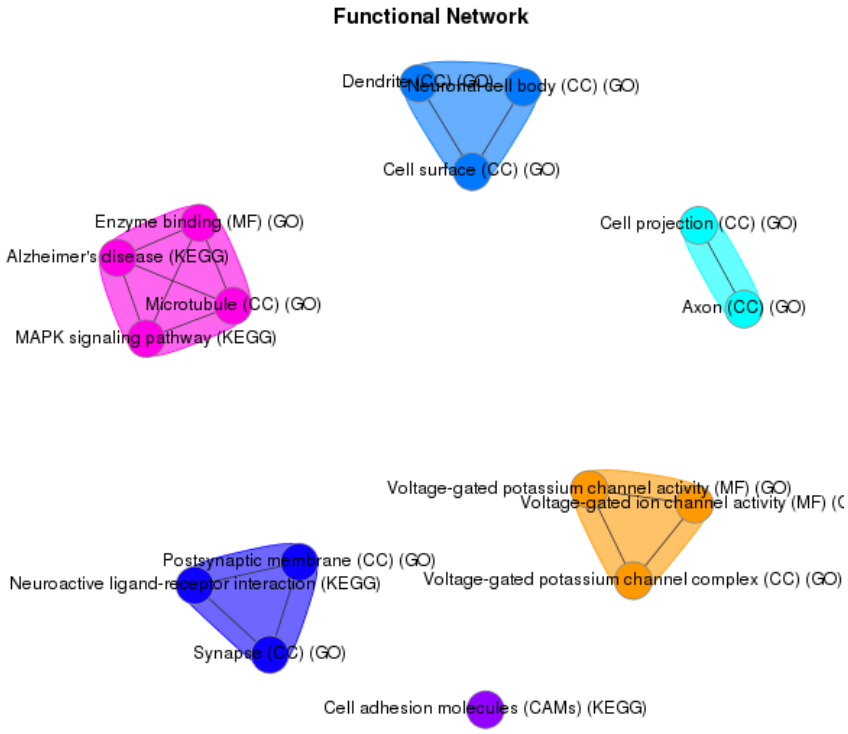
By default, the functional network is built establishing links between nodes (genes or terms) in the same gene-term sets. Depending on the tool, this network might have few or no edges:

```
functionalNetwork(incidMatTerms,  weighted=TRUE, plotOutput="dynamic")
```

To plot a network with links between all the terms in the same cluster or metagroups, use `fea2incidMat()` with the `$cluster` or `$metagroup` slots from the FEA, in order to consider the whole cluster/metagroup as a gene-term set:

```
incidMatTerms <- fea2incidMat(feaAlzheimer$metagroups, clusterColumn="Metagroup",
    key="Terms",
    filterAttribute="Silhouette.Width", filterThreshold=0.2)
functionalNetwork(incidMatTerms, legendText=FALSE, plotOutput="dynamic")
```

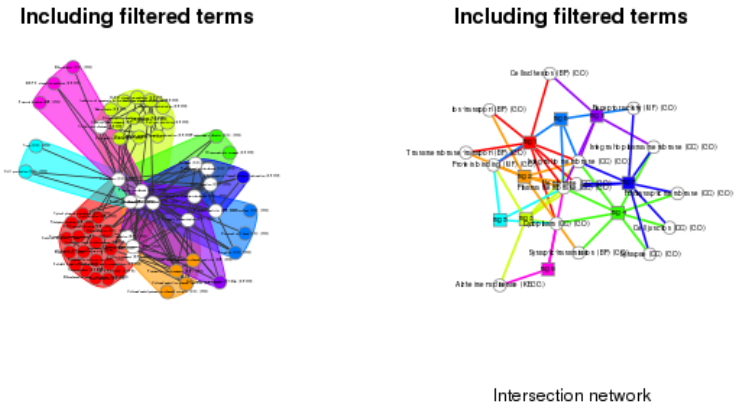
```
functionalNetwork(incidMatTerms, legendText=FALSE)
```



Since GeneTerm Linker filters out generic and redundant terms from the final metagroups, by default these terms are not plotted. To include them in the graph, set the argument `removeFiltered=FALSE` (only available for GeneTerm Linker).

```
incidMatTerms <- fea2incidMat(feaAlzheimer, key="Terms", removeFilteredGtl=FALSE)
par(mfrow=c(1,2))
functionalNetwork(incidMatTerms, vLabelCex=0.2,
    plotTitle="Including filtered terms", legendText=FALSE)
functionalNetwork(incidMatTerms, plotType="bipartite", vLabelCex=0.4,
    plotTitle="Including filtered terms")
```

For more information on the filtered terms see (Fontanillo et al) or `http://gtlinker.cnb.csic.es/gtset/help` .



4.5 Genes - Terms networks

To build a genes-terms network, we can use the bipartite plot with the appropriate formatting of the input matrices.
For many FEA tools it will be enough with applying the `fea2incidMat()` directly to the `$geneTermSets` matrix selecting the gene-term sets we want to plot. i.e. gene-term sets in a specific cluster, filter generic terms (terms annotated to more than X genes), etc...
Note that this approach might not be appropriate for GeneTerm Linker, since it groups several terms into each gene-term set.

```
txtFile <- paste(file.path(system.file('examples', package='FGNet')),
    "DAVID_Yeast_raw.txt", sep=.Platform$file.sep)
feaResults_David <- format_david(txtFile, jobName="David_example",
    geneLabels=genesYeast)
```

```
feaResults_David <- fea_david(names(genesYeast), email="...",
    geneLabels=genesYeast)
```

```
gtSets <- feaResults_David$geneTermSets
gtSets <- gtSets[gtSets$Cluster %in% c(9),]
gtSets <- gtSets[gtSets$Pop.Hits<500,]
```

Then, create a terms-genes incidence matrix with `fea2incidMat()`, and plot the network...

```
termsGenes <- t(fea2incidMat(gtSets, clusterColumn="Terms")$clustersMatrix)
library(R.utils)
rownames(termsGenes) <- sapply(strsplit(rownames(termsGenes), ":"),
    function(x) capitalize(x[length(x)]))
termsGenes[1:5,1:5]
```

```
##                                                CDC16 DOC1 GLC7
## Anatomical structure morphogenesis                 1    1    1
## Cell differentiation                               1    1    1
```

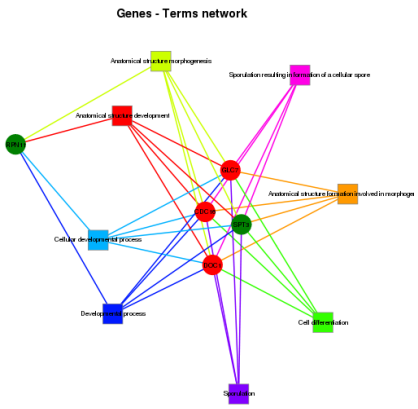
```
## Sporulation resulting in formation of a cellular spore     1    1    1
## Developmental process                                      1    1    1
## Sporulation                                                1    1    1
##                                                         RPN11 SPT3
## Anatomical structure morphogenesis                          1    1
## Cell differentiation                                        0    1
## Sporulation resulting in formation of a cellular spore      0    1
## Developmental process                                       1    1
## Sporulation                                                 0    1
```

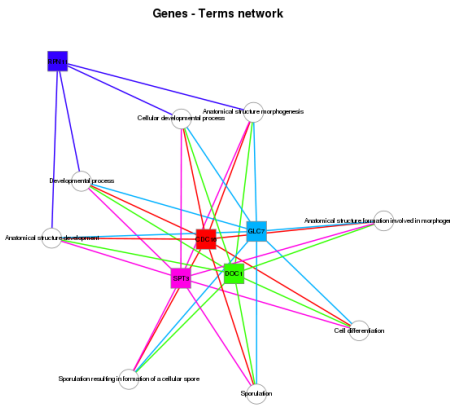
Network with genes colored based on their expression and terms on alphabetical order:

```
functionalNetwork(t(termsGenes), plotType="bipartite", keepAllNodes=TRUE,
    legendPrefix="", plotTitle="Genes - Terms network", plotTitleSub="",
    geneExpr=genesYeastExpr, plotExpression="Fill")
```

Network with genes colored by alphabetical order (from red to pink), terms white:

```
functionalNetwork(termsGenes, plotType="bipartite", keepAllNodes=TRUE,
    legendPrefix="", plotTitle="Genes - Terms network", plotTitleSub="")
```



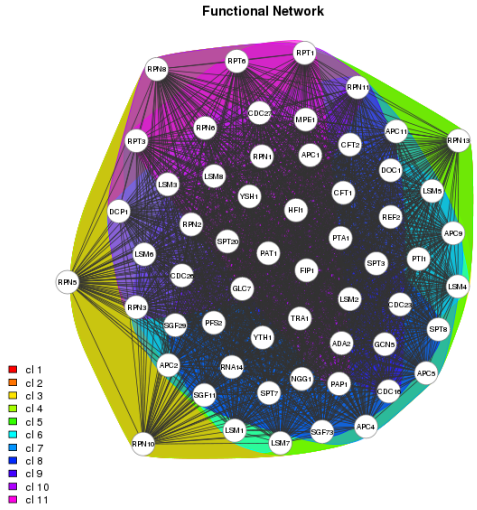


5 Filtering and selecting clusters

As an example of analysis of a network with very overlapping clusters, we will use the yeast gene list analyzed with DAVID:

```
incidMat <- fea2incidMat(feaResults_David)
functionalNetwork(incidMat)
```

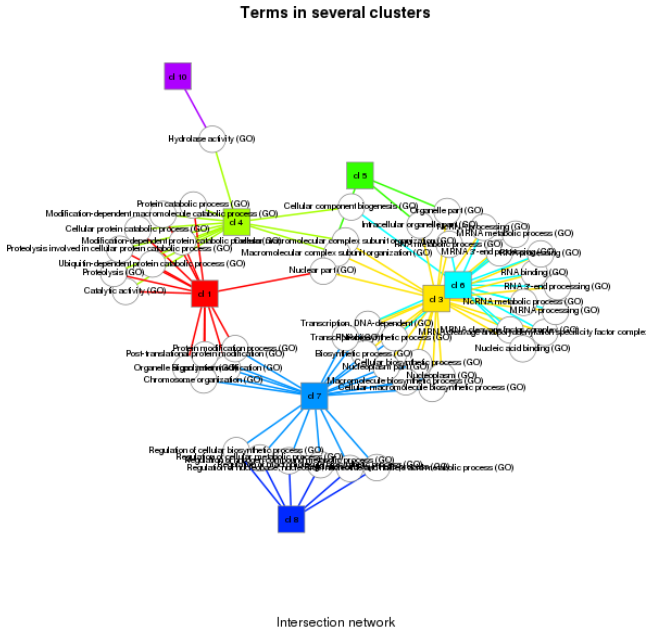
FGNet 23




```
incidMatTerms <- fea2incidMat(feaResults_David, key="Terms")
```

```
functionalNetwork(incidMatTerms$clustersMatrix, plotOutput="dynamic",
  weighted=TRUE, eColor="grey")
```

```
functionalNetwork(incidMatTerms$clustersMatrix, plotType="bipartite",
 plotTitle="Terms in several clusters")
```



5.1 Filtering based on a *cluster* property

The clusters to plot can be selected/filtered based on any property that is available in the clusters matrix:

```
colnames(feaResults_David$clusters)
```

```
## [1] "Cluster"                "nGenes"
## [3] "ClusterEnrichmentScore" "Genes"
## [5] "Terms"                  "keyWordsTerm"
```

i.e. Selecting the clusters with highest Enrichment Score or least genes (setting `eColor=NA`, plots the networks without edges):

```
par(mfrow=c(1,2))

# Highest enrichment score
filterProp <- as.numeric(as.character(feaResults_David$
    clusters$ClusterEnrichmentScore))
quantile(filterProp, c(0.10, 0.25, 0.5, 0.75, 0.9))
```

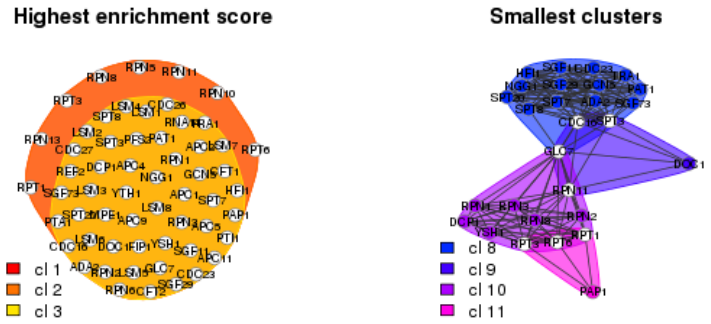
```
##        10%        25%        50%        75%        90%
## 0.08585003 0.33812100 5.90148600 7.65222050 7.85874500
```

```
incidMatFiltered <- fea2incidMat(feaResults_David,
    filterAttribute="ClusterEnrichmentScore",
    filterOperator="<", filterThreshold=7.65)
functionalNetwork(incidMatFiltered, eColor=NA,
    plotTitle="Highest enrichment score")

# Lowest genes
quantile(as.numeric(as.character(feaResults_David$clusters$nGenes)),
    c(0.10, 0.25, 0.5, 0.75, 0.9))
```

```
##  10%  25%  50%  75%  90%
##  5.0 13.5 44.0 55.0 58.0
```

```
incidMatFiltered <- fea2incidMat(feaResults_David,
 filterAttribute="nGenes", filterOperator=">", filterThreshold=20)
functionalNetwork(incidMatFiltered, plotTitle="Smallest clusters")
```

To use any property that is not available in the `$clusters` data frame, just add it as column to the dataframe.



5.2 Selecting clusters with specific keywords

```
keywordsTerm(getTerms(feaResults_David), nChar=100)
```

```
##                              Cluster 1                              Cluster 2
## "Cellular protein catabolic process"                  "Metabolic process"
##                              Cluster 3                              Cluster 4
##                         "Transcription" "Cellular protein catabolic process"
##                              Cluster 5                              Cluster 6
##                            "Organelle"                    "MRNA processing"
##                              Cluster 7                              Cluster 8
##                         "Transcription" "Regulation of biosynthetic process"
##                              Cluster 9                             Cluster 10
##   "Anatomical structure development"                 "Hydrolase activity"
##                             Cluster 11
##                           "ATP binding"
```

```
keywords <- c("hydrolase")
selectedClusters <- sapply(getTerms(feaResults_David),
    function(x)
    any(grep(paste("(", paste(keywords, collapse="|") ,")",sep=""), tolower(x))))
```

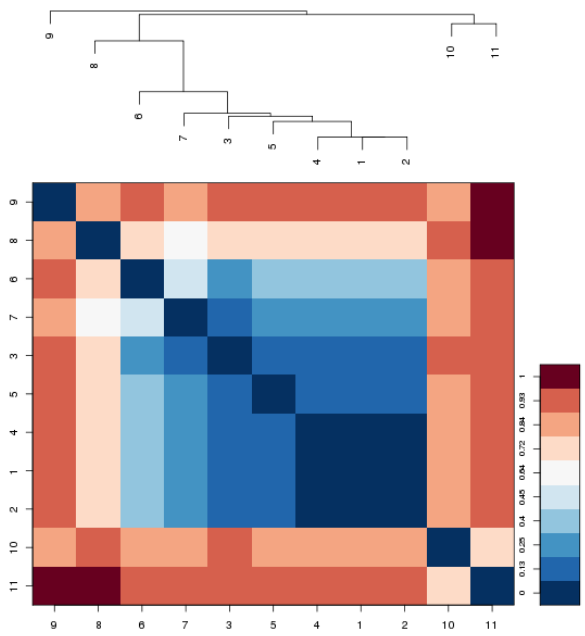
```
getTerms(feaResults_David)[selectedClusters]
```

```
tmpFea <- feaResults_David
tmpFea$clusters <- cbind(tmpFea$clusters, keywords=selectedClusters)
incidMatSelection <- fea2incidMat(tmpFea,
 filterAttribute="keywords", filterOperator="!=",filterThreshold="TRUE")
functionalNetwork(incidMatSelection, plotType="bipartite")
```

5.3 Selecting specific clusters

`clustersDistance()` allows to explore the overlap between clusters:

```
distMat <- clustersDistance(incidMat)
```



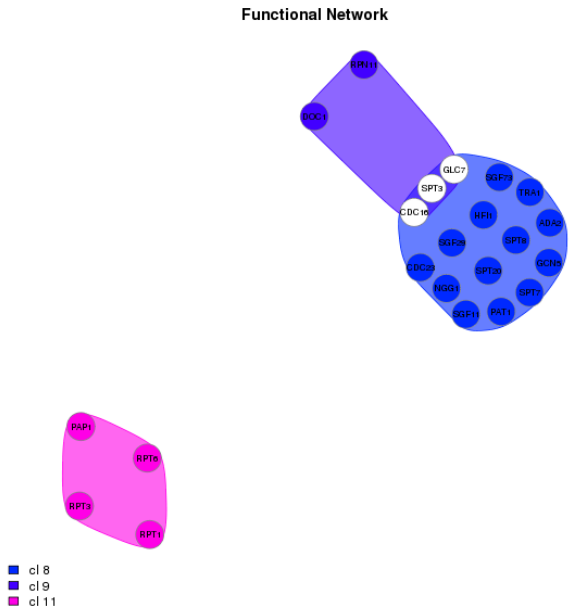
Clusters 4, 1 and 2 seem to be completely overlapping (distance 0). While cluster 11 does not have any intersection with clusters 8 and 9. Let's see:

```
selectedClusters <- rep(FALSE, nrow(feaResults_David$clusters))
selectedClusters[c(8,9,11)] <- TRUE

tmpFea <- feaResults_David
tmpFea$clusters <- cbind(tmpFea$clusters, select=selectedClusters)
incidMatSelection <- fea2incidMat(tmpFea,
  filterAttribute="select", filterOperator="!=",filterThreshold="TRUE")
functionalNetwork(incidMatSelection, eColor=NA)
```

5.4 Filtering based on a *gene-term set* property

In some ocasions, it might also be useful to filter out gene-term sets within a cluster. i.e. The terms in the top of the GO ontologies are annotated to many genes and make most clusters overalp.
To filter out terms, (1) filter or select the terms in the the `feaReults$geneTermSets` data frame, (2) save it as text file, and (3) import it with `readGeneTermSets()`
In this case, we will use DAVID's example, and keep the terms thar are annotated to less than 100 genes in yeast:

FGNet 27




```
# Same analysis, setting overlap to 6:
feaResults_David_ov6 <- fea_david(names(genesYeast), geneLabels=genesYeast,
    email="example@email.com",
    argsWS=c(overlap=6, initialSeed=3, finalSeed=3, linkage=0.5, kappa=50))
```

```
# Filter/select
sum(feaResults_David_ov6$geneTermSets$Pop.Hits < 100)
```

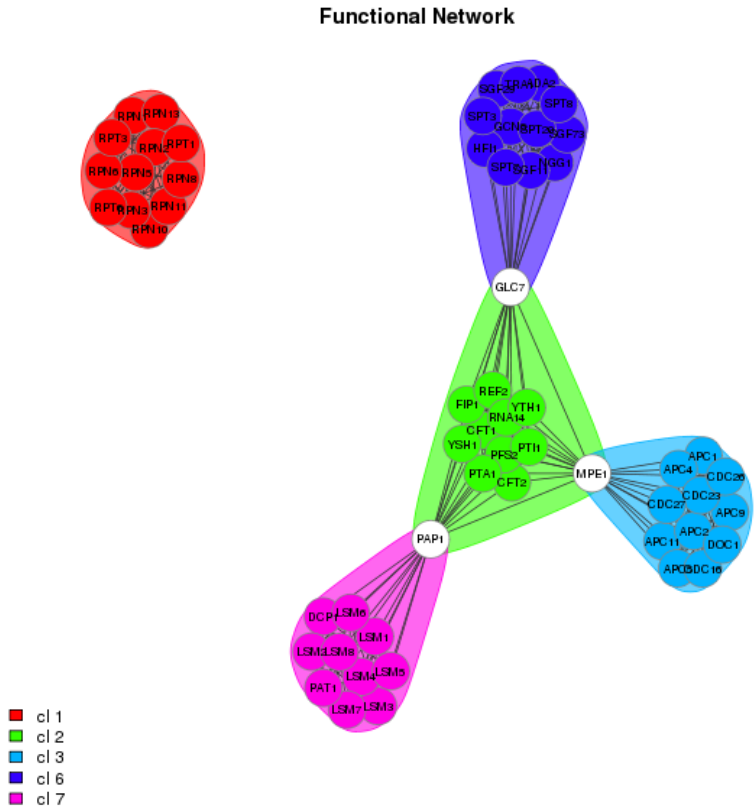
```
## [1] 64
```

```
gtSets <- feaResults_David_ov6$geneTermSets[
    feaResults_David_ov6$geneTermSets$Pop.Hits < 100,]
# Save
write.table(gtSets, file="david_filteredGtsets.txt", sep="\t",
    col.names = TRUE, quote=FALSE)
# Load with "readGeneTermSets"
feaResults_filteredGtsets <- readGeneTermSets("david_filteredGtsets.txt",
    tool="DAVID")
# ...
functionalNetwork(fea2incidMat(feaResults_filteredGtsets))
```

To explore the distribution of genes-terms in a specific organism:

```
# Yeast
library(org.Sc.sgd.db)
goGenesCountSc <- table(sapply(as.list(org.Sc.sgdGO2ORF), length))
barplot(goGenesCountSc, main="Number of genes annotated to GO term (Sc) ",
```

FGNet 28



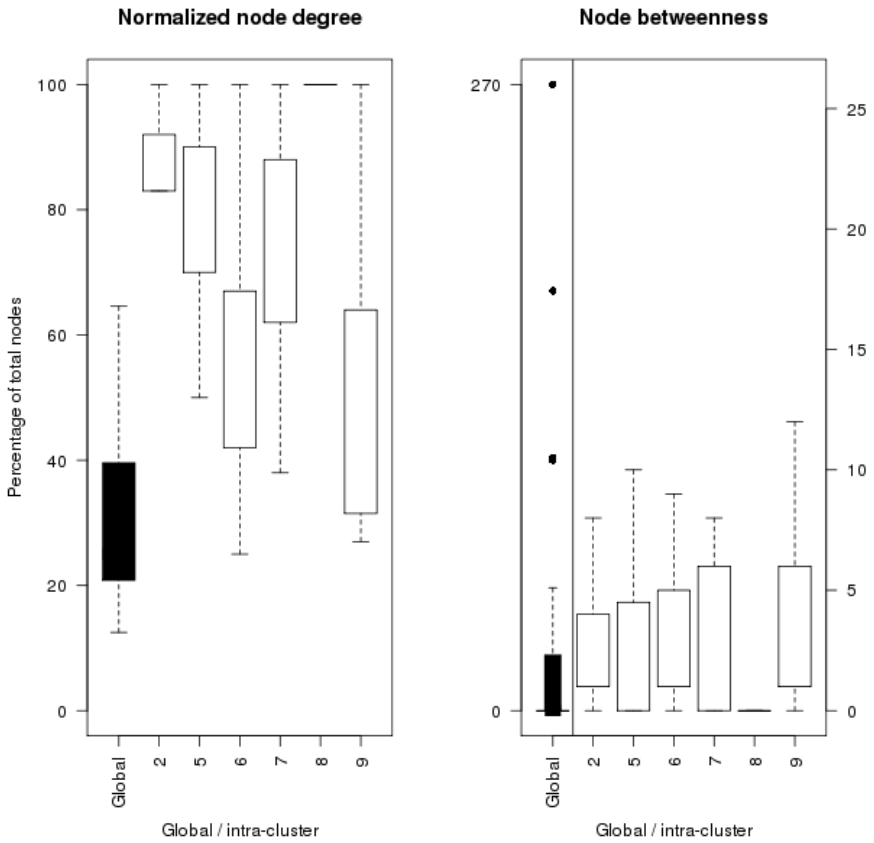

```
    xlab="Number of genes", ylab="Number of GO terms")

# Human
library(org.Hs.eg.db)
goGenesCountHs <- table(sapply(as.list(org.Hs.egGO2EG), length))
barplot(goGenesCountHs, main="Number of genes annotated to GO term (Human)",
    xlab="Number of genes", ylab="Number of GO terms")
```

6 Other auxiliary functions

6.1 analyzeNetwork()

`analyzeNetwork()` can be used to explore the structure of the network. It also returns statistics about the nodes betweenness within each cluster, etc...
The example with GeneTerm Linker (Alzheimer):

```
incidMatFiltered <- fea2incidMat(feaAlzheimer,
    filterAttribute="Silhouette Width", filterOperator="<", filterThreshold=0.2)
stats <- analyzeNetwork(incidMatFiltered)
```



```
names(stats)
```

```
## [1] "degree"           "betweeness"       "transitivity"
## [4] "betweenessMatrix" "hubsList"         "intraHubsCount"
```

```
stats$transitivity
```

```
## commonClustersNw   commonGtSetsNw
##        0.6947699        0.5943638
```

`$degree` and `$betweenness` are the values used for the plots. They contain the values for each of the nodes in the global network (commonClusters) and within each cluster/metagroup (subsets of commonGtSets network). The degree is given as percentage, normalized based on the total number of nodes of the network. i.e. a value of 90 in a network of 10 nodes, would mean the actual degree of the node is 9: it is connected to 9 nodes (90% of 10)).
The betweenness of each node in each cluster as matrix:

```
head(stats$betweennessMatrix)
```

Inter-modular hubs: Nodes with betweenness within the top 75% in the global network

```
stats$hubsList$Global
```

```
## [1] "NLGN4X"  "HCN1"    "CHRM1"   "CNTNAP1" "SCN2A"
```

Intra-modular hubs: Nodes with betweenness within the top 75% in each cluster sub-network

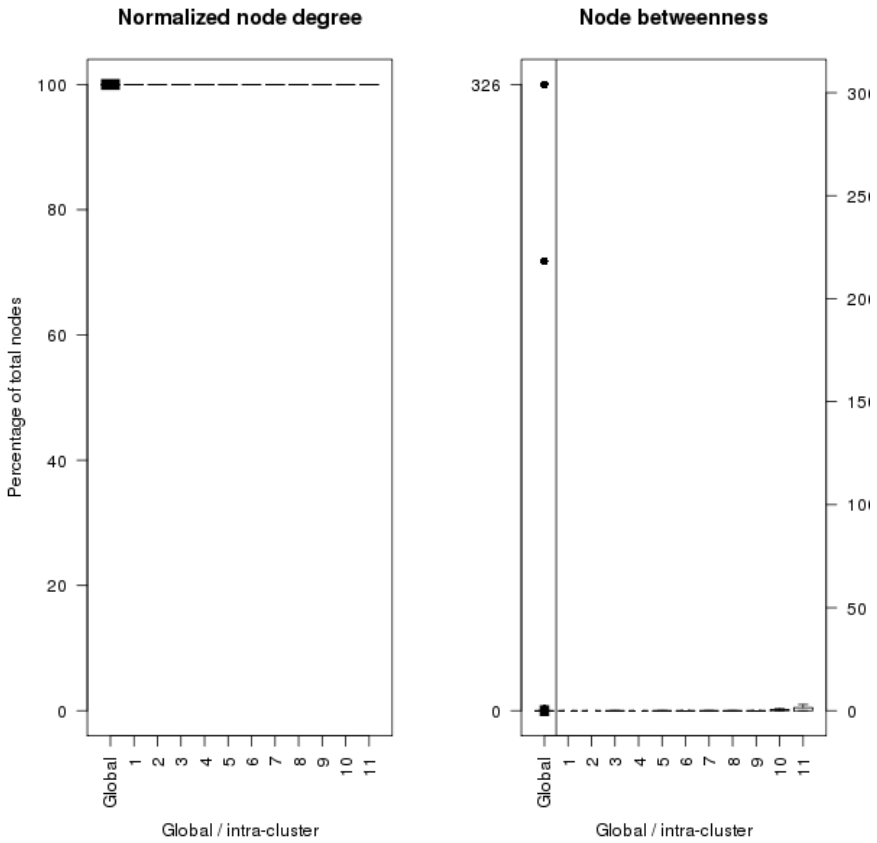
```
stats$hubsList$"9"
```

```
## [1] "MAPT"  "CALM3" "APOE"
```

FGNet 31

DAVID's example:

```
incidMat_metab <- fea2incidMat(feaResults_David)
analyzeNetwork(incidMat_metab)
```

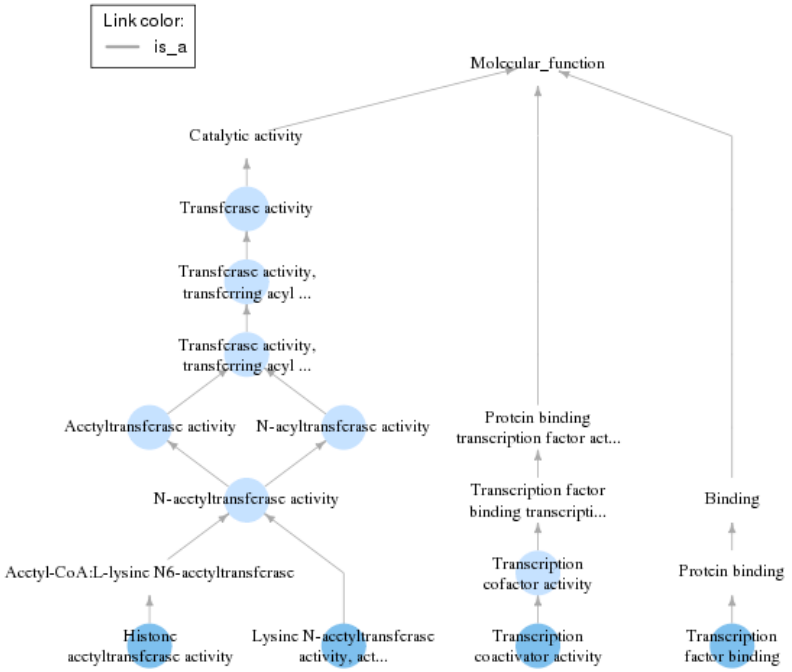


Note the structure of the network varies not only depending on the dataset, but also on the tool. Since tools like DAVID link all the nodes/terms within each cluster, their internal normalized degree is always 100%.

6.2 plotGoAncestors()

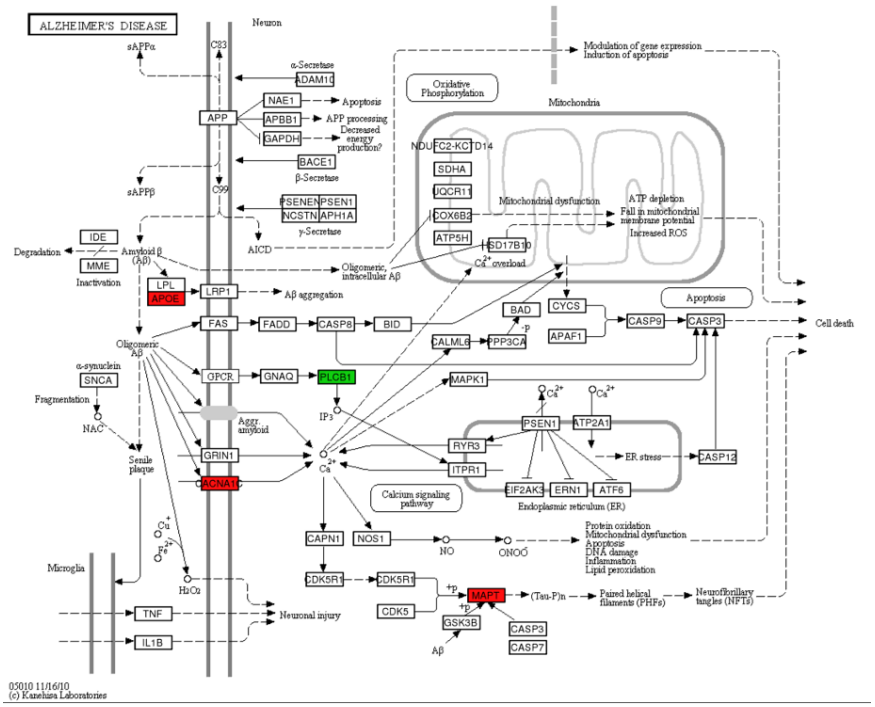
`plotGoAncestors()` and `plotKegg()` also allow to explore the significant gene term sets:

```
goIds <- getTerms(feaResults_David, returnValue="GO")[[7]]
plotGoAncestors(goIds, ontology="MF", nCharTerm=40)
```



6.3 plotKegg()

```
keggIds <- getTerms(feaAlzheimer, returnValue="KEGG")[[3]]
plotKegg("hsa05010", geneExpr=genesAlzExpr, geneIDtype="GENENAME")
# Saved as .png in current directory
```



Acknowledgements

This work was supported by Instituto de Salud Carlos III [Research Projects PS09/00843 and PI12/00624] and by a grant from the Junta de Castilla y Leon and the European Social Fund to S.A and C.D.

References

[1] Huang DW, Sherman BT, Lempicki RA. *Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources.* Nature Protoc. 2009;4(1):44-57.

[2] Huang DW, Sherman BT, Lempicki RA. *Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists.* Nucleic Acids Res. 2009;37(1):1-13.

[3] Fontanillo C, Nogales-Cadenas R, Pascual-Montano A, De Las Rivas J (2011) *Functional Analysis beyond Enrichment: Non-Redundant Reciprocal Linkage of Genes and Biological Terms.* PLoS ONE 6(9): e24289. doi:10.1371/journal.pone.0024289

[4] Alexa A, and Rahnenfuhrer J (2010) topGO: Enrichment analysis for Gene Ontology. R package version 2.16.0. URL: `http://www.bioconductor.org/packages/release/bioc/html/topGO.html`

[5] Luo W, Friedman MS, Shedden K, Hankenson KD, Woolf PJ (2009) GAGE: generally applicable gene set enrichment for pathway analysis. BMC Bioinformatics. 10:161. URL: `http://www.bioconductor.org/packages/release/bioc/html/gage.html`